MySQL Document Store

Daisuke Inagaki/ 稲垣　大助

(daisuke.Inagaki@oracle.com)

MySQL Global Business Unit

MySQL Principal Sales Consultant

Oracle Corporation Japan.

Safe Harbor Statement

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

The world's most popular open source database

世界で最も普及しているオープンソース データベース

ORACLE

MySQL 8.0

SQL & NoSQL
Unmatched Flexibility

Continuous Delivery Model
Launchpad for Future

New Architecture
Eliminating Legacy

Open Source & Enterprise
Scalable & Robust

ORACLE

MySQL：Webアプリケーション開発効率向上を実現

Mobile Friendly

位置情報ベースのサービス向けの機能強化と絵文字を含めたユニコード対応😃

Developer First

ハイブリッド型のデータモデルとアクセスAPIによる開発柔軟性

Data Driven

アプリケーションデータ分析による運用中サービス改良支援

24x7 at Scale

Scalable & Stable

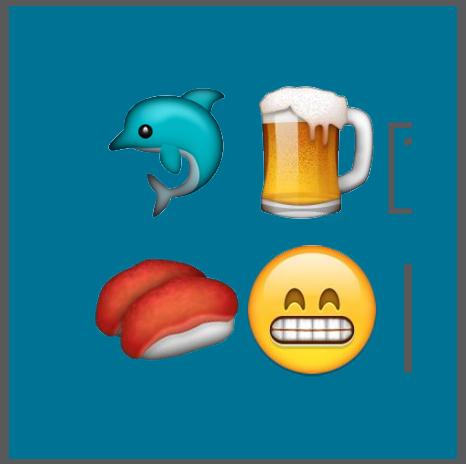
アクセス集中時の処理改良、セキュリティと耐障害性強化

ORACLE

MySQL：モバイルアプリとの親和性

GIS(空間図形情報)サポートの強化

- 位置情報ベースのサービスとの連携の改良
- MySQL 5.7 にて Boost.Geometry ライブライリーを統合
- MySQL 8.0 にて球面座標と測地座標系(SRS)サポート



ユニコードをデフォルトキャラクタセットに

- 絵文字をサポートする **`utf8mb4` がデフォルトのキャラクタセットに**
- ユニコード文字列の処理性能が16倍以上向上するケースも
- Unicode 9.0 をサポート
- UCA(Unicode照合アルゴリズム)ベースの新しい各言語用の照合

MySQL：アプリケーション開発者に柔軟性を

データ型

JSON データ型

リレーショナルなテーブルと非構造データとシームレスに統合。さらに MySQL 8.0 では更新性能の最適化

SQL 関数

JSON 関数

JSON データの参照更新のための各種 SQL 関数を実装。MySQL 8.0 では JSON データを SQL で分析するための変換関数も追加

ハイブリッドAPI

MySQL X DevAPI

SQL と CRUD な NoSQL のハイブリッドAPIによる開発柔軟性

ドキュメントストア概要

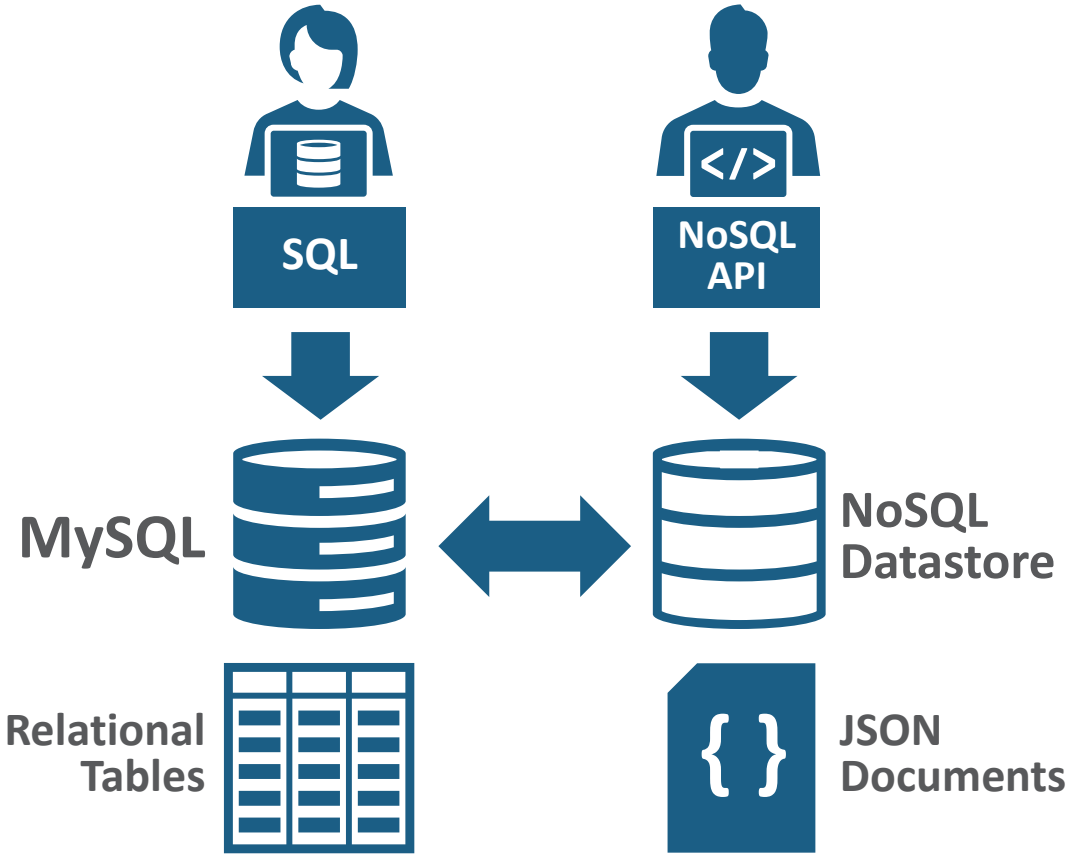
Relational Databases

- データの整合性
  - 正規化
  - 制約(**foreign keys,….**)
- **Atomicity, Consistency, Isolation, Durability**
- 原子性、一貫性、独立性、永続性
  - **ACID compliant**
  - トランザクション
- **SQL**
  - 強力なクエリ言語

NoSQL or Document Store

- スキーマレス
  - スキーマ設計不要、正規化不要、外部KEYやデータ型なし....
- 柔軟なデータ構造
  - 埋め込み型の配列やオブジェクト
  - ありふれたデータを最適化し、リレーショナルモデルでは表現できない場合の有効な解決策
- JSON
  - フロントエンドに近い
  - JSネイティブ
  - 分かりやすい

RDBMSとNoSQLデータストアを併用する際の懸念事項



開発者

複数のAPIを学習する必要がある

データ管理

テーブルとJSONドキュメントの確実なデータ同期が困難

運用

個別に運用ツールを導入して別々の運用管理が求められる

DBMS or NoSQL ?

ドキュメントストア機能拡張

- リレーショナル、スキーマレスを同じ技術スタックで利用可能 
- MySQLに実装されている機能を活用可 (レプリケーション,InnoDB[ACID]等)
- JSONデータ型と関数,追加されたCRUD APIによる容易な開発

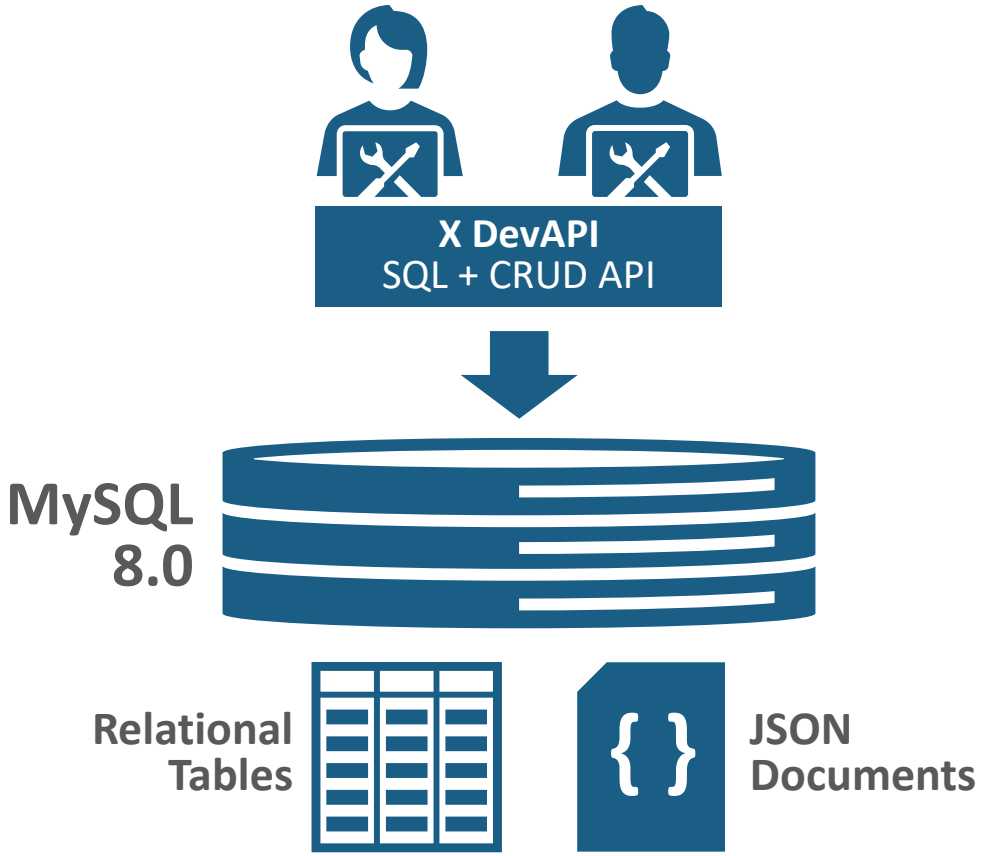
【ステークスホルダーのニーズを満たす為の機能追加】

開発チーム:
[x] スキーマレス
[x] 迅速なプロトタイプ/シンプルAPI
[x] ドキュメントモデル

オペレーション:
[x] パフォーマンス管理/可視化
[x] 堅牢レプリケーション,バックアップ, リストア
[x] 包括的なツールによるエコシステム

ビジネス:
[x] データを確実に保護 = ACIDトランザクション
[x] 全てのデータをキャプチャー = 拡張性/スキーマレス
[x] スケージュール/製品化の時間 = 迅速なサービス開発

MySQL Document Store: SQL + NoSQL = MySQL



開発者にとっての柔軟性

統合されたAPIによる柔軟性

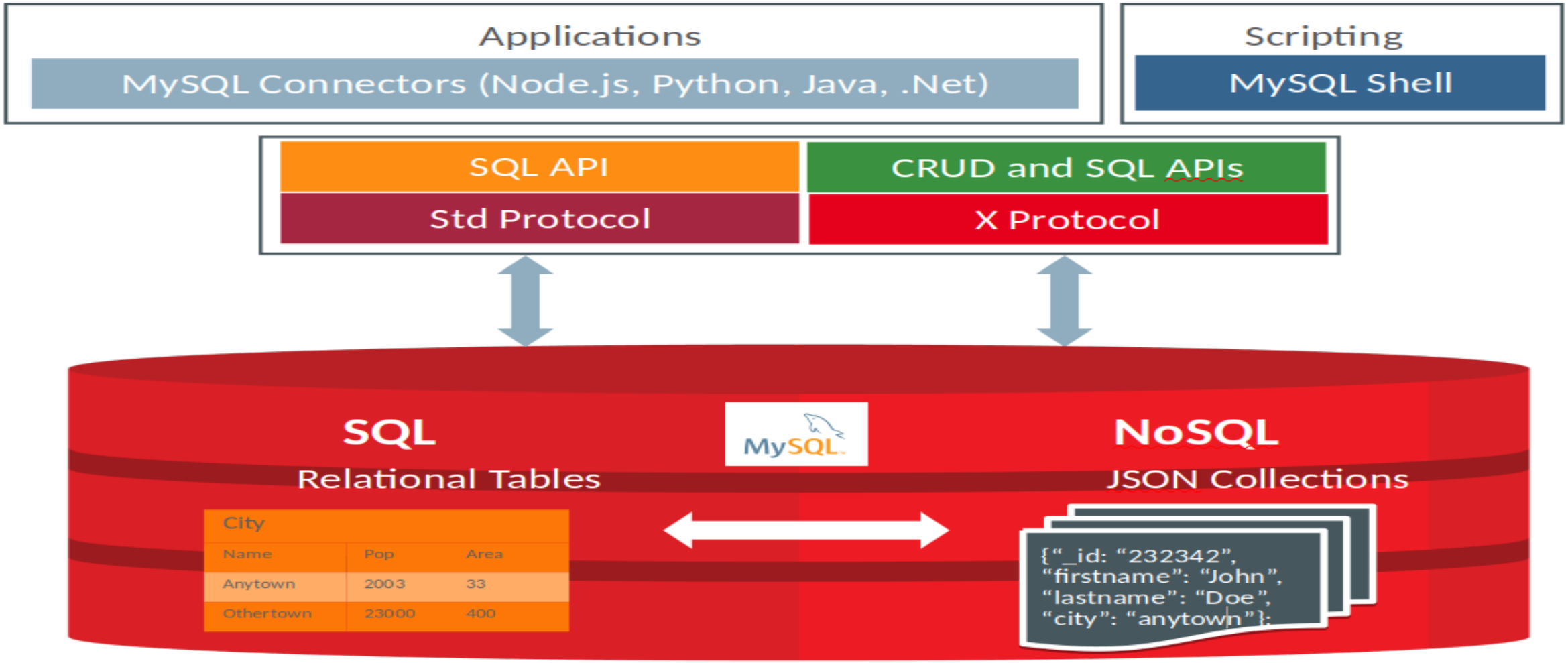
データ管理の信頼性と柔軟性

単一のデータストアなのでデータ同期不要 & テーブルとJSONドキュメントのJOINも可能

運用効率の向上

単一データベースのみの運用で済むので管理負荷低減

MySQL Document Store



X Protocol

X Protocol

The X Protocol focuses on:
- extensibility
- performance
- security

- 非同期APIサポート – 並列処理とバッチ処理をサポート
  - パイプライン方式 – 複数リクエストを送信，ラウンドトリップを削減
  - CRUD ==大量に小さなPKを処理, 独立した複数のクエリーを処理
- ミドルウエアとの親和性
  - ルーティング、シャーディング、読み取り書き込みスプリッティング　(XSESSION)
- オープンスタンダードの利用: TLS (Transport Layer Security), SASL(Simple Authentication and Security Layer), Protobuf (Protocol Buffers)等

```
INSTALL PLUGIN mysqlx SONAME 'mysqlx.so';
```

```
+-------------+----------------+--------------------+
| PLUGIN_NAME | PLUGIN_VERSION | PLUGIN_DESCRIPTION |
+-------------+----------------+--------------------+
| mysqlx      | 1.0            | X Plugin for MySQL |
+-------------+----------------+--------------------+
```

詳細: http://mysqlserverteam.com/mysql-5-7-12-part-2-improving-the-mysql-protocol/

X DevAPI

X Plugin (MySQL) ⇔ X Protocol ⇔ X DevAPI (Driver)

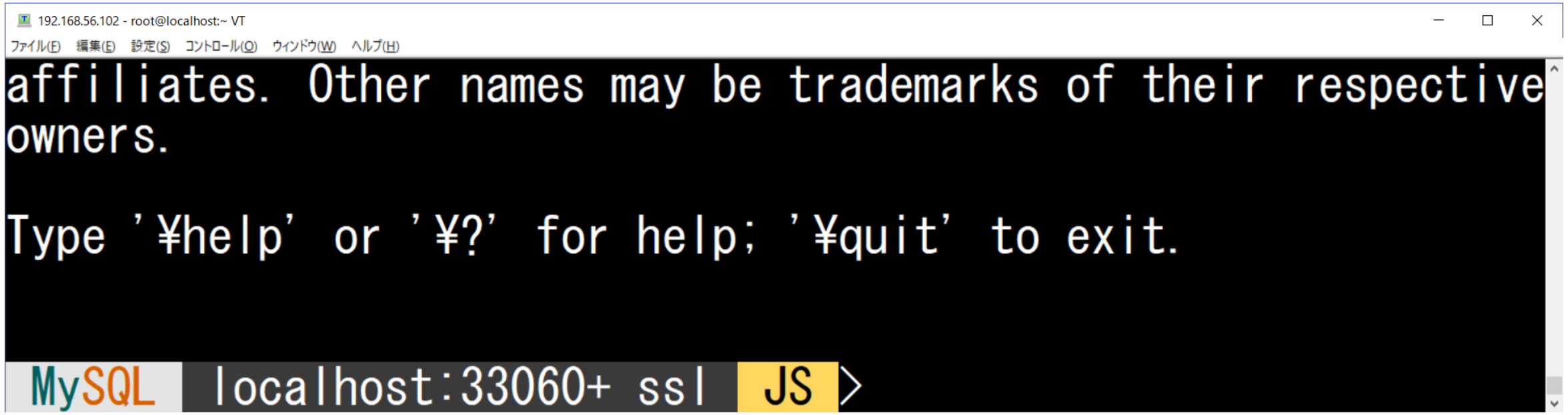
- X Pluginを有効にする事で、X Protocol経由で通信可能
- ドキュメントとテーブルのコレクションに対してのCRUD処理
- NoSQLライクな構文でドキュメントに対しCRUD処理可能
- Fluent API

  - ✓ MySQL Connector/node.js (1.0.x)
  - ✓ MySQL Connector/J (6.0.x)
  - ✓ MySQL Connector/Net (7.0.x)
  - ✓ MySQL Connector/python(2.2.x)
  - ✓ MySQL Shell (1.0.x)

```
prod = sess.getSchema("prod")
res = prod.users.
    find("$.name = 'Milk'").
    fields(["name", "properties"])
```

参照：http://dev.mysql.com/downloads/connector/

新しいインタフェース MySQL Shell



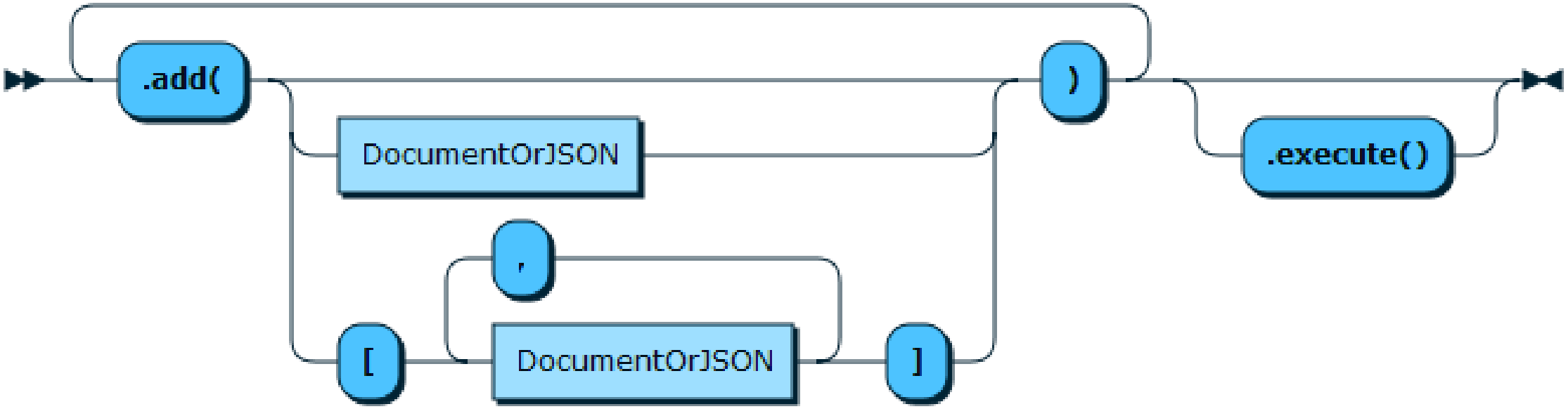
MySQL Shell Info

```
MySQL  localhost:33060+  docstore  JS  \s
MySQL Shell version 8.0.11

Session type:                   X
Connection Id:                  13
Default schema:
Current schema:                 docstore
Current user:                   root@localhost
SSL:                            Cipher in use: DHE-RSA-AES128-GCM-SHA256 TLSv1.2
Using delimiter:                ;
Server version:                 8.0.11 MySQL Community Server - GPL
Protocol version:               X protocol
Client library:                 8.0.11
Connection:                     localhost via TCP/IP
TCP port:                       33060
Server characterset:            utf8mb4
Schema characterset:            utf8mb4
Client characterset:            utf8mb4
Conn. characterset:             utf8mb4
Uptime:                         3 hours 59 min 26.0000 sec
```

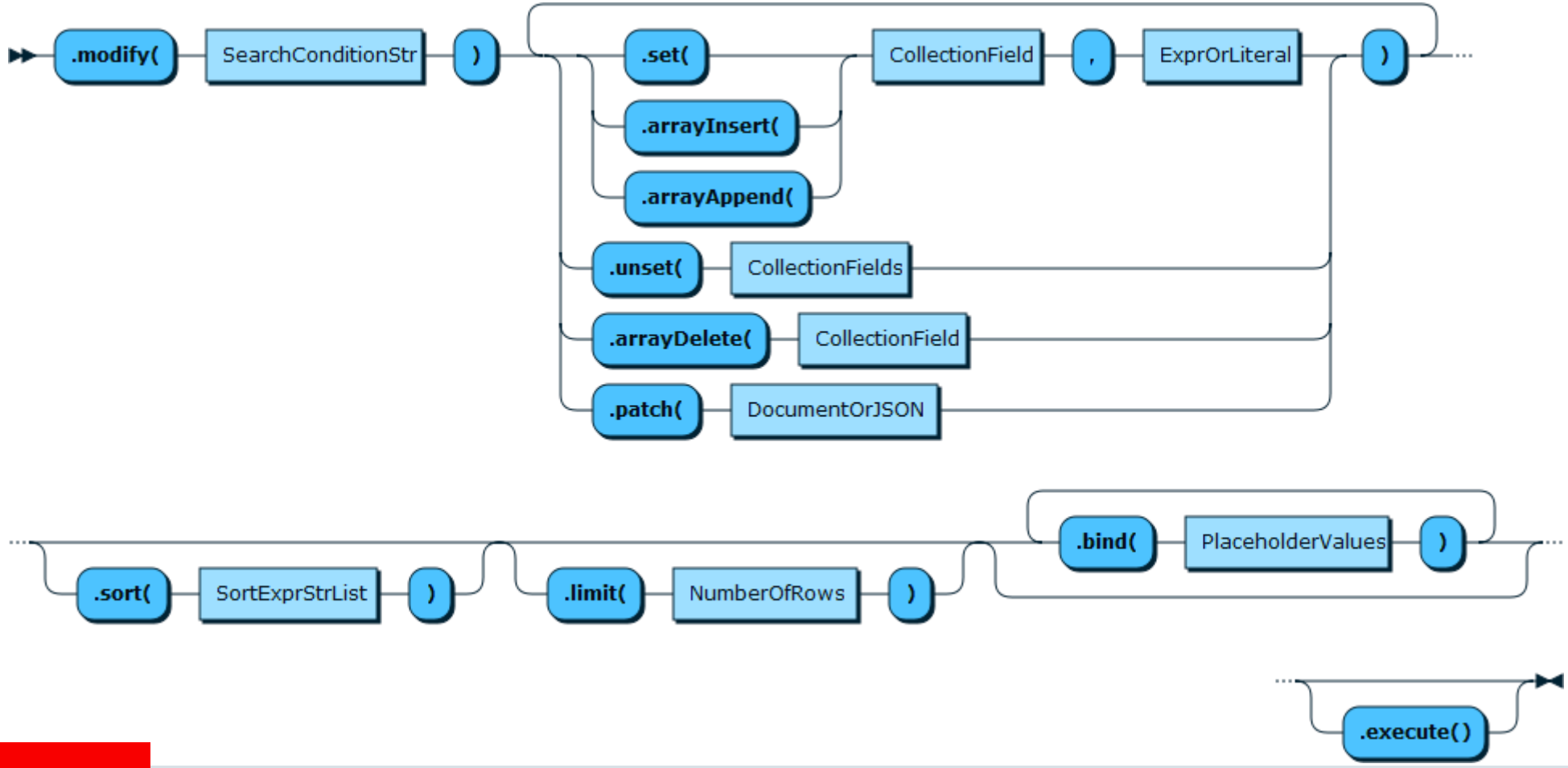
CRUD operations for collections

Add a document



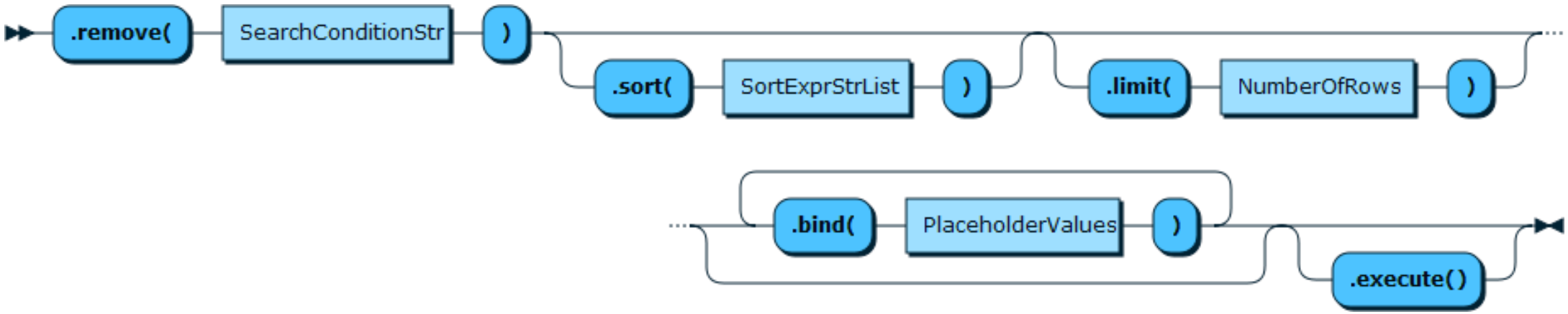
CRUD operations for collections

Modify a document

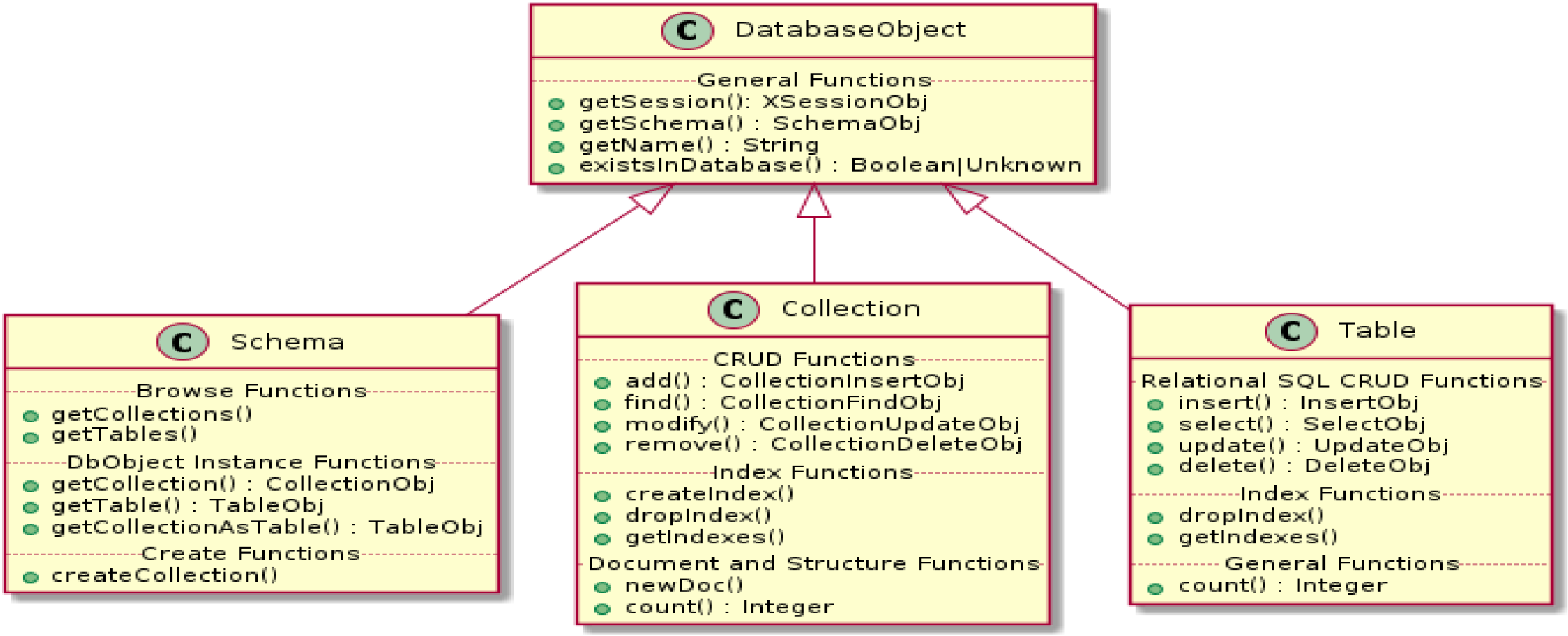


CRUD operations for collections

Remove a document



MySQL Document Store Objects



SQL

JSON Functions

MySQL 5.7 and 8.0

JSON_ARRAY_APPEND()

JSON_ARRAY_INSERT()

JSON_ARRAY()

JSON_CONTAINS_PATH()

JSON_CONTAINS()

JSON_DEPTH()

JSON_EXTRACT()

JSON_INSERT()

JSON_KEYS()

JSON_LENGTH()

JSON_MERGE[_PRESERVE]()

JSON_OBJECT()

JSON_QUOTE()

JSON_REMOVE()

JSON_REPLACE()

JSON_SEARCH()

JSON_SET()

JSON_TYPE()

JSON_UNQUOTE()

JSON_VALID()

JSON_PRETTY()

JSON_STORAGE_SIZE()

JSON_STORAGE_FREE()

JSON_ARRAYAGG()

JSON_OBJECTAGG()

JSON_MERGE_PATCH()

JSON_TABLE()

JSON_REMOVE()

JSONドキュメントからデータを削除し、結果を返します。

```
[NEW57]> select * from T_JSON_DOC;
+----+-------------------------------------------------------------------------------------------------+------------+
| id | body                                                                                            | price_only |
+----+-------------------------------------------------------------------------------------------------+------------+
|  1 | {"id": 1, "name": "自転車", "price": 10000, "Conditions": ["NEW", 2015, "Excellent"]}           |   10000.00 |
|  2 | {“id”: 2, “name”: “テレビ”, “price”: 30000, “Conditions”: [“USED”, 2013, “故障”]}              |   30000.00 |
|  3 | {"id": 3, "name": "冷蔵庫", "price": 17131, "Conditions": ["NEW", 2015]}                        |   17131.00 |
|  4 | {“id”: 4, “name”: “オートバイ”, “price”: 500000, “Conditions”: [“NEW”, 2015]}                  |  500000.00 |
|  5 | {"id": 5, "name": "自転車", "price": 25000, "Quantity": "[1,10]", "Conditions": ["NEW", 2015]}  |   25000.00 |
+----+-------------------------------------------------------------------------------------------------+------------+
```

```
[NEW57]> select JSON_REMOVE(body,"$.Quantity") from T_JSON_DOC where id = 5;
+-----------------------------------------------------------------------+
| JSON_REMOVE(body,"$.Quantity")                                        |
+-----------------------------------------------------------------------+
| {"id": 5, "name": "自転車", "price": 25000, "Conditions": ["NEW", 2015]}  |
+-----------------------------------------------------------------------+
```

```
[NEW57]> update T_JSON_DOC set body = JSON_REMOVE(body,"$.Quantity") where id = 5;
```

JSON_SET ()

JSONドキュメントにデータを挿入または更新し、結果を返します。

replaces existing values and adds nonexisting values.

```
[NEW57]> select * from T_JSON_DOC;
+----+-------------------------------------------------------------------------------+------------+
| id | body                                                                          | price_only |
+----+-------------------------------------------------------------------------------+------------+
|  1 | {“id”: 1, “name”: “自転車”, “price”: 10000, “Conditions”: [“NEW”, 2015]}      |   10000.00 |
|  2 | {"id": 2, "name": "テレビ", "price": 30000, "Conditions": ["USED",2013]}      |   30000.00 |
|  3 | {"id": 3, "name": "冷蔵庫", "price": 11154, "Conditions": ["NEW", 2015]}      |   11154.00 |
|  4 | {"id": 4, "name": "冷蔵庫", "price": 50000, "Conditions": ["NEW", 2015]}      |   50000.00 |
|  5 | {"id": 5, "name": "自転車", "price": 25000, "Conditions": ["NEW", 2015]}      |   25000.00 |
+----+-------------------------------------------------------------------------------+------------+
```

```
[NEW57]> SELECT JSON_SET(body,'$.name',"オートバイ", '$.price',500000) from T_JSON_DOC where id = 4;
+-------------------------------------------------------------------------------+
| JSON_SET(body,'$.name',"オートバイ", '$.price',500000)                        |
+-------------------------------------------------------------------------------+
| {"id": 4, "name": "オートバイ", "price": 500000, "Conditions": ["NEW", 2015]} |
+-------------------------------------------------------------------------------+
```

```
[NEW57]> update T_JSON_DOC set body = JSON_SET(body,'$.name',"オートバイ", '$.price',500000) where id = 4;
```

JSON_INSERT ()

JSONドキュメントにデータを挿入し、結果を返します

inserts values without replacing existing values.

```
[NEW57]> select * from T_JSON_DOC;
+----+-------------------------------------------------------------------------+------------+
| id | body                                                                    | price_only |
+----+-------------------------------------------------------------------------+------------+
|  1 | {“id”: 1, “name”: “自転車”, “price”: 10000, “Conditions”: [“NEW”, 2015]}      |   10000.00 |
|  2 | {"id": 2, "name": "テレビ", "price": 30000, "Conditions": ["USED",2013]}      |   30000.00 |
|  3 | {"id": 3, "name": "冷蔵庫", "price": 11154, "Conditions": ["NEW", 2015]}      |   11154.00 |
|  4 | {"id": 4, "name": "冷蔵庫", "price": 50000, "Conditions": ["NEW", 2015]}      |   50000.00 |
|  5 | {"id": 5, "name": "自転車", "price": 25000, "Conditions": ["NEW", 2015]}      |   25000.00 |
+----+-------------------------------------------------------------------------+------------+
```

```
[NEW57]> select JSON_INSERT(body,'$.Quantity','[1,10]') from NEW57.T_JSON_DOC where id = 5;
+-----------------------------------------------------------------------------------------+
| JSON_INSERT(body,'$.Quantity','[1,10]')                                                 |
+-----------------------------------------------------------------------------------------+
| {"id": 5, "name": "自転車", "price": 25000, "Quantity": "[1,10]", "Conditions": ["NEW", 2015]}     |
+-----------------------------------------------------------------------------------------+
```

```
[NEW57]> update NEW57.T_JSON_DOC set body = JSON_INSERT(body,'$.Quantity','[1,10]') where id = 5;
```

JSON_REPLACE ()

JSON文書の既存の値を置き換え、その結果を返します。

replaces only existing values.

```
[NEW57]> select * from T_JSON_DOC;
+----+-------------------------------------------------------------------------------+------------+
| id | body                                                                          | price_only |
+----+-------------------------------------------------------------------------------+------------+
|  1 | {“id”: 1, “name”: “自転車”, “price”: 10000, “Conditions”: [“NEW”, 2015]}      |   10000.00 |
|  2 | {“id”: 2, “name”: “テレビ”, “price”: 30000, “Conditions”: [“USED”,2013]}      |   30000.00 |
|  3 | {"id": 3, "name": "冷蔵庫", "price": 11154, "Conditions": ["NEW", 2015]}      |   11154.00 |
|  4 | {"id": 4, "name": "冷蔵庫", "price": 50000, "Conditions": ["NEW", 2015]}      |   50000.00 |
|  5 | {"id": 5, "name": "自転車", "price": 25000, "Conditions": ["NEW", 2015]}      |   25000.00 |
+----+-------------------------------------------------------------------------------+------------+
```

```
[NEW57]> select JSON_REPLACE(body,"$.price",FLOOR(10000 + (RAND() * 9000))) from T_JSON_DOC where id = 3;
+-------------------------------------------------------------------------+
| JSON_REPLACE(body,"$.price",FLOOR(10000 + (RAND() * 9000)))             |
+-------------------------------------------------------------------------+
| {"id": 3, "name": "冷蔵庫", "price": 18359, "Conditions": ["NEW", 2015]} |
+-------------------------------------------------------------------------+
```

```
[NEW57]> update T_JSON_DOC set body = JSON_REPLACE(body,"$.price",FLOOR(10000 + (RAND() * 9000))) where id = 3;
```

Generated Columns (生成列)とは?

ファンクショナル・インデックス

生成列を使用して列を作成
対象:JSONデータから情報を抽出
(例)　User ID, 製品ID, サービスID等

```
[NEW57]> CREATE TABLE `T_JSON` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `feature` json NOT NULL,
  `feature_type` varchar(30) GENERATED ALWAYS AS (json_unquote(feature->"$.type")) VIRTUAL,
  `feature_street` varchar(30) GENERATED ALWAYS AS (json_extract(`feature`,'$.properties.STREET')) VIRTUAL,
  PRIMARY KEY (`id`),
  KEY `idx_feature_street` (`feature_street`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4;
```

```
[NEW57]> alter table features add index idx_feature_type(`feature_type`);
Query OK, 0 rows affected (6.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

生成列に対し,オンラインでインデックスを追加。
→ 高速なJSONデータの検索が可能に!!

インデックス利用で処理が 1.25秒 → 0.06秒

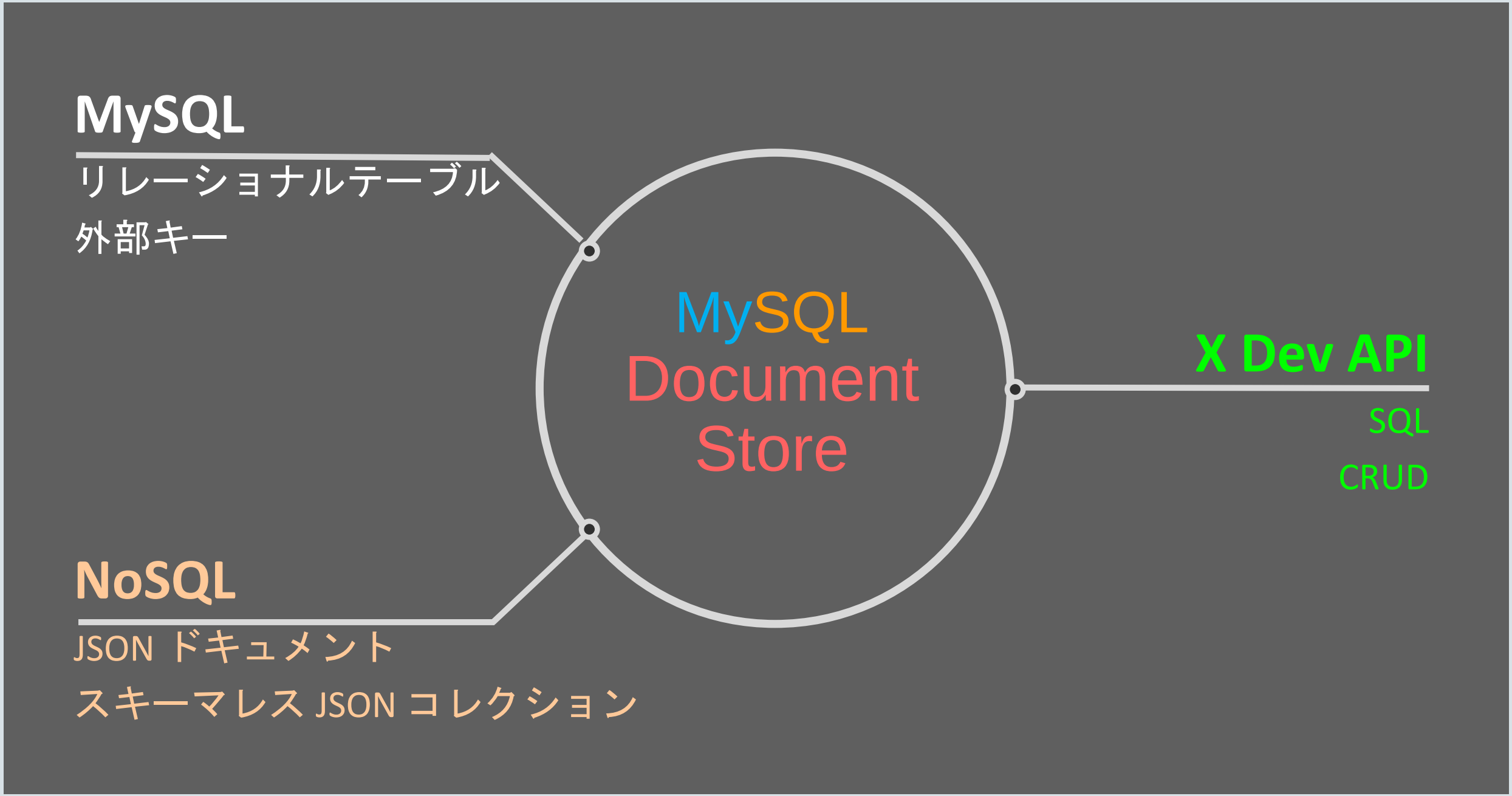
```
[NEW57]> explain select distinct(feature_type) from features;
+----+-------------+----------+------------+-------+-----------------+-----------------+---------+------+--------+----------+-------------+
| id | select_type | table    | partitions | type  | possible_keys   | key             | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+----------+------------+-------+-----------------+-----------------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | features | NULL       | index | idx_feature_type | idx_feature_type | 123     | NULL | 199013 |   100.00 | Using index |
+----+-------------+----------+------------+-------+-----------------+-----------------+---------+------+--------+----------+-------------+
```

生成列に設定可能なインデックスオプション

Advantage
Disadvantage

Generated Column (STORED) 挿入・更新時に演算、値を格納 データも含まれる為、参照が早い	Generated Column (VIRTUAL) 参照時に演算、値は格納しない メタデータ変更のみ、INSERT/UPDATEが早い
Primary and Secondary	Secondary Only
BTREE, Fulltext, GIS	BTREE Only
Mixed with fields	Mixed with virtual column only
Requires table rebuild	No table rebuild
Not Online	INSTANT Alter
	Faster Insert

Bottom Line: 主キー, FULLTEXTまたは仮想GISインデックスを必要とする場合を除き,デフォルトのVIRTUALで問題無い。


MySQL
リレーショナルテーブル
外部キー
NoSQL
JSON ドキュメント
スキーマレス JSON コレクション
MySQL
Document
Store
X Dev API
SQL
CRUD

DBMS or NoSQL ?


NoSQL + SQL =
MySQL

Integrated Cloud

Applications & Platform Services


ORACLE®